Algorithms - Exercise 10

Due Wednesday 12/1 24:00

1. Give an algorithm that for a given integer c returns its integer square root, if exists, and prompts otherwise. The running time should be polynomial in the input length.

2. Let n_1, n_2, n_3 be $3, 5, 8$ respectively.

   (a) What is the Chinese remainders representation (modulo n_1, n_2, n_3) of the number 99?

   (b) What is the number who's Chinese remainders representation is $(2, 4, 0)$?

   Write you intermediate calculations.

3. Prove that if Alice's public exponent e is 3 and Eve obtains Alice's secret exponent d, then Eve can factor Alice's modulus n in time polynomial in the number of bits of n.

4. Alice is using RSA cryptosystem with exponents e, d and modulus n. Use the fact that RSA is multiplicative, that is

$$m_1^e m_2^e \equiv (m_1 m_2)^e \mod \text{n}$$

   to prove that if Eve has a procedure that can efficiently decrypt 1 percent of messages randomly chosen from Z_n and encrypted by Alice, then Eve can employ a probabilistic algorithm to decrypt every message encrypted by Alice with high probability.

5. Suppose that there is a polynomial time algorithm, A, that given a prime number p and a number $0 \leq c \leq p-1$, computes one of the square roots of (c mod p) if ($c = x^2$ mod p) for some x, and answers "no" if it is not of this form. When p is not a prime number there is no guarantee on the behavior of A. (You may be interested to know that such an algorithm indeed exists). We now use A to develop an algorithm that tests whether a given number is prime. Consider the following algorithm, given y:

   - Choose a random $0 \leq z \leq y-1$.
   - Compute $gcd(z, y)$, if it is not 1 output "not prime".
   - Compute $c = z^2$ mod y, and run the algorithm A on z.
   - If A returns z or $y - z$ then output "prime". Otherwise output "not prime".

   Prove:

   (a) The algorithm runs in polynomial-time (in the input length).

   (b) If y is prime then with probability 1 the algorithm gives the right answer. If y is not a prime then with probability at least 1/2 the algorithm gives the right answer.

   (c) For any constant $0 < \delta < 1$ (as small as you want) there is a probabilistic polynomial-time algorithm that solves the problem of primality testing and makes an error with probability at most δ.